Lucas Spangher ljs14

1 Definitions

What definitions are necessary?

2 Lecture

2.1 Bin Packing

Recall "Bin Packing" from last week:

1. We have n items, $x_1, x_2, ..., x_n$, where x_i denotes the size of item i. The size of all items, $x_i < 1$.
2. We have m bins, $b_1, b_2, ..., b_m$ where b_i denotes the capacity of bin i. All capacities are 1 (?).

How can we use space most efficiently? I.e. how can we put all of our items in bins while filling the least amount of bins possible?

Lemma: At most 1 bin $\leq \frac{1}{2}$ full.

Proof: Say we pack the bins such that 2 bins are less than half full. We can then combine these two without breaching the capacity of either one. Using this logic recursively, we can consolidate all bins that are less than half into other bins such that only one bin is packed $\leq \frac{1}{2}$.

In the way this is formulated, it is discrete (there are finitely many items.) Let's make this a continuous question:

Say we have k types of items, and each one is pf size $\geq \epsilon$. Each i^{th} type is of ∞ supply. If the best case occurred, i.e. each item was size ϵ, then we be able to fit floor($\frac{1}{\epsilon}$) in each bin. Therefore, there are at most $k^{\frac{1}{\epsilon}}$ configurations of items that can fit in one box.

If you have n bins, how many items can you store?

—-

To come up with the answer, let's think about it in algorithm form. How do we produce an instance such that the number of items we have is small? We answer this with an approximation algorithm. First, think of items with around the same size as being the same. We can formalize this as picturing the items as lying on the continuum (0,1).

1. Remove all items of size $< \epsilon$. (?)
2. Create $\frac{1}{\epsilon^2}$arbitrary buckets on this line such that each bucket has $n\epsilon^2$ items. See Figure 1.

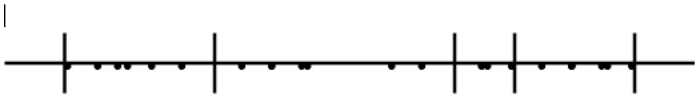

Figure 1: Representation of the bucketing.

3. Round the size of the every item to the upper bound of its bucket. Call this rounding the OPT_u. Also, imagine another instance where the size of each item is rounded to the lower bound of its bucket. Call this solution OPT_l.

   What is the difference between OPT_u and OPT_l? Notice the following:

   (a) The $n\epsilon^2$items of the highest bucket on OPT_u go to 1, and the $n\epsilon^2$items of the lowest bucket go to 0. Thus, $OPT_l + n\epsilon^2 \geq OPT_u$ (? Shouldn't this be equal?)

   (b) $OPT_l \leq OPT$, i.e. the optimal solution without rounding has to be at least the solution with rounding.

   (c) $OPT \geq n\epsilon$.

$$OPT_u \leq OPT_l + n\epsilon^2 \leq OPT + n\epsilon^2 \leq (1+\epsilon)OPT \tag{1}$$

   The final inequality is due to the (3) note above: if $OPT \geq n\epsilon$, then $OPT + n\epsilon^2 \leq n\epsilon + n\epsilon^2 = (1+\epsilon)n\epsilon^2 = (1+\epsilon)OPT$.

   If there are $\frac{1}{\epsilon^2}$ items, then there are at most $\left(n\frac{1}{\epsilon}\right)$ configurations.

4. Use FIRST-FIT (? defined in an earlier lecture notes?) on the small items. The total volume that you leave unfilled is at most ϵ. Thus, since you are using $1-\epsilon$ of the volume of each bin, the space complexity is $1+O(\epsilon)$. This is constant with respect to the problem.

What type of problem is this? It is not FPTAS, because FPTAS must be polynomial in $\frac{1}{\epsilon}$ and all inputs. However, this is PTAS. It is not the best we can do. The algorithm can get $OPT + O(\log OPT * \log\log OPT)$. (Recall that PTAS $= OPT + O(OPT)$ and FPTAS $= OPT + \theta(OPT)$. Another way of expressing this is that the algorithm is solvable in $O(n^{f(k,\frac{1}{\epsilon}}$.

A proof for this is the following:

1. Find the number of intergral solutions of $x_1 + ...x_k = \frac{1}{\epsilon}$ where $x_i \geq 0$.

2. Counting the number of solutions is equivalent to counting the number of configurations, which is equivalent to counting the number of ways of placing $\frac{1}{\epsilon}$ balls, which, finally, can be expressed as the number of binary strings with $\frac{1}{\epsilon}$ 0's and $(k-1)$ 1's.

3. Thought of in this way, it is easier to see that there $\binom{\frac{1}{\epsilon}+k-1}{\frac{1}{e}}$. Call this value M.

4. If M is the number of configurations and n is the number of bins, the number of solutions to $x_1 + ... + x_M = n$ is $\binom{M+n-1}{n} \leq \binom{M+n}{n} \leq (M+n)^M$. This is roughly equivalent to $n^{f(\frac{1}{\epsilon},k)}$.

Thus, this is PTAS.

2.2 LP Rounding

2.2.1 Vertex Cover

First, recall VERTEX COVER: If we have a graph G, then what is the minimum subset of vertices we can pick such that every each has at least 1 vertex in the subset? Formally:

$$\begin{aligned} G &= (V, E) \\ min(S) &\in V \\ s.t. \forall e_{(u,v)} &\in E \\ u \in S &\;||\; v \in S \end{aligned}$$

Formulated in an LP, this looks like the following (x_v can be thought of as the probability of vertex v being chosen in the subset):

$$min(\sum_{v \in S} x_v)$$
$$x_u + x_v \geq 1 \; \forall (u,v) \in E$$
$$x_v \geq 0 \; \forall v \in G$$

If these are integral 0,1, than this is NP-hard.

Let's say you get some fraction solution, and you lose a factor α in converting to the integral solution.

$$x^* \rightarrow x^*_{int} \tag{2}$$

Than, $x^*_{int}(v) \leq \alpha \sum_{v \in S} x^*_v$. This is an α-approximate solution. A simple rounding implementation of the vertex cover algorithm is: let $x_v = 1$ if $x_v \geq \frac{1}{2}$. Otherwise, let $x_v = 0$. This satisfies the original LP formulation, because for any pair of vertices $(u,v) \in E$, at least one must be $\geq \frac{1}{2}$, otherwise the statement $x_u + x_v \geq 1 \forall (u,v) \in E$ would not be fulfilled. We also see that this is a 2-approximation, because the worst case is that $x_v = \frac{1}{2} \forall v$, meaning that every value would double.

2.2.2 Set Cover

Set cover is very similar to vertex cover. However, instead of finding a minimum set of vertices that includes at least one vertex for every edge, we are finding a minimum cost of the subset of edges such that every subset has at least one vertex covered. U is the universe of all sets.

Let's take the LP for Set Cover.

$$min(\sum_{s \in S} c_s x_s) \qquad \text{i.e. minimize the total cost of S}$$

$$\sum_{s \in S} x_s \ \forall e \in U, x_s \geq 0$$

$$s \in U$$

A rounding of this is the following: if x^* is an optimal fractional solution, then let $x^*_{int}(s) = 1$ with probability x^*_s and let $x_{int} *_s = 0$ with probability 1- x^*_s. Thus, $\mathbb{E}(x^*_{int}(s) = x^*(s)$, so the approximation is a 1-approximation.

However, this may not be a feasible solution, because it is possible that in rounding down to 0, we allow that not all subsets are covered. So, employ the following fix, which is almost the same as repeating the former $\log n * s$: let $x_i nt^*(s) = 1$ with probability $x^*(s) \log n$ and 0 otherwise. The probability that element e is covered is:

$$1 - \prod_{s \in S} (1 - x^*(s) \log n = 1 - \frac{1}{e^{\log n}} = 1 - \frac{1}{n} \tag{3}$$

If an element is not covered, use it's cheapest covering set to cover it:

$$\mathbb{E}(cost) = \log n \sum_{s \in S} c_s x^*_s + n * \frac{1}{n} \tag{4}$$

It turns out that the cheapest cover is at least the optimal solution; i.e. $OPT \geq min_{s \forall e} c_s : e \in s$.

Note: Set cover is an important test for computational techniques. Every technique gives an $O(\log n)$ solution for set cover.